

Pemrograman Berorientasi Obyek

Class Diagram

anton@ukdw.ac.id



What is UML ?

- Sebuah bahasa **pemodelan** terstandar untuk bahasa pemrograman berorientasi obyek
- UML merupakan perpaduan dari tiga paradigma pemodelan sistem object oriented:
 - OMT (James Rumbaugh)
 - OOSE (Ivan Jacobson)
 - Booch (Grady Booch)



What is UML ?

- UML **bukan alat untuk membuat software**, UML membantu membuat **model** dari software yang akan dibuat
- UML berupa gambar, berisi notasi-notasi untuk membuat software blueprints (**rancangan**)
- Digunakan juga untuk **mendokumentasikan**



Why is UML Important

- UML untuk membuat software **blueprints** bagi *analysts, designers dan programmers*
- UML merupakan bahasa “universal” untuk pihak-pihak yang terlibat dalam pembuatan suatu software
- Pada OO, modelling merupakan bagian yang sangat penting. UML membantu proses modelling tersebut



What is UML goals ?

- Menyediakan seperangkat notasi dan tools untuk melakukan pemodelan software dengan sederhana dan mudah digunakan
- UML bersifat independen dari bahasa pemrograman tertentu, UML bukan bahasa pemrograman, UML hanya berupa notasi untuk pemodelan software



Who is need UML ?

- Semua pihak yang terlibat dalam pembuatan software
- System Analysts, Programmers, Business Designer, dsb
- UML berupa notasi-notasi yang berupa gambar sehingga mudah untuk dipelajari, selain itu UML juga tidak terkait pada bahasa pemrograman tertentu



UML Diagrams

- Tipe diagram pada UML dibagi menjadi 2 kategori :
- **Structure Diagrams**
Diagram yang menggambarkan aspek **statis** dari system yang kita buat
 - Contoh: Class Diagram
- **Behavior Diagrams**
Diagram yang menggambarkan aspek **dinamis** dari system yang kita buat
 - Contoh: Activity Diagram, Sequence Diagram



Diagrams in the UML

- Class diagram
- Object diagram
- Use case diagram
- Sequence diagram
- Collaboration diagram
- Statechart diagram
- Activity diagram
- Component diagram
- Deployment diagram

Class Diagram

- Apa yang ada pada class diagram
 - Class dan interface beserta atribut dan method-nya
 - Relasi yang terjadi antar objek
 - Constraint terhadap objek-objek yang saling berhubungan
 - Inheritance untuk organisasi class yang lebih baik

Class Diagram

- A class diagram consists of three sections:
 - The upper part holds the name of the class
 - The middle part contains the attributes of the class
 - The bottom part gives the methods or operations the class can take or undertake

Tools

- **Free Solution:**
- Dia (Gnome)
- IBM Rational Modeler
- Kivio (KDE)
- Netbeans
- Umbrello UML Modeller (KDE)
- StarUML
- ArgoUML

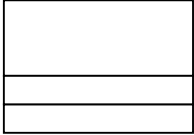

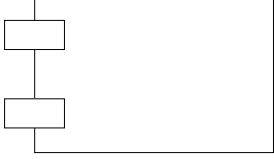
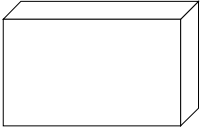


Tools

- **Proprietary Solution :**
- Microsoft Visio
- Rational Software Architects
- Enterprise Architects
- Poseidon for UML

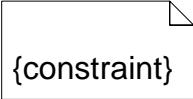


Structural Modeling: Core Elements

Construct	Description	Syntax
class	a description of a set of objects that share the same attributes, operations, methods, relationships and semantics.	
interface	a named set of operations that characterize the behavior of an element.	
component	a modular, replaceable and significant part of a system that packages implementation and exposes a set of interfaces.	
node	a run-time physical object that represents a computational resource.	

Structural Modeling: Core Elements


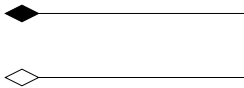
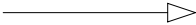
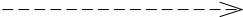
(cont'd)

Construct	Description	Syntax
constraint ¹	a semantic condition or restriction.	

¹ An extension mechanism useful for specifying structural elements.

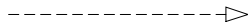
Structural Modeling:

Core Relationships

Construct	Description	Syntax
association	a relationship between two or more classifiers that involves connections among their instances.	
aggregation	A special form of association that specifies a whole-part relationship between the aggregate (whole) and the component part.	
generalization	a taxonomic relationship between a more general and a more specific element.	
dependency	a relationship between two modeling elements, in which a change to one modeling element (the independent element) will affect the other modeling element (the dependent element).	

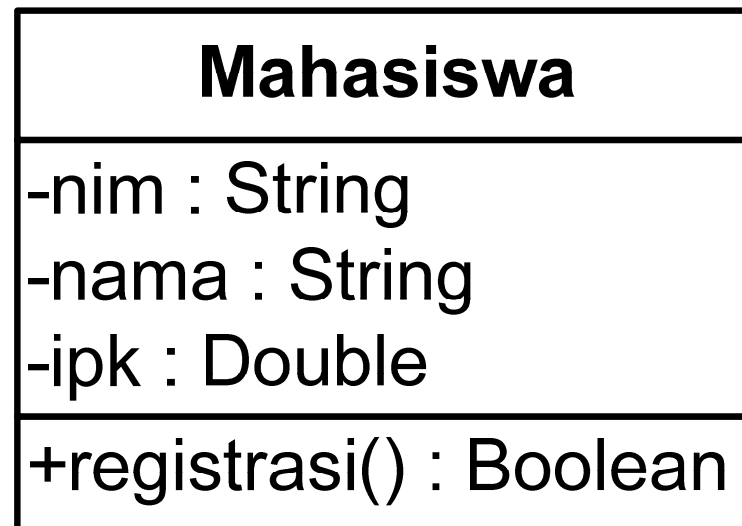
Structural Modeling:

Core Relationships (cont'd)

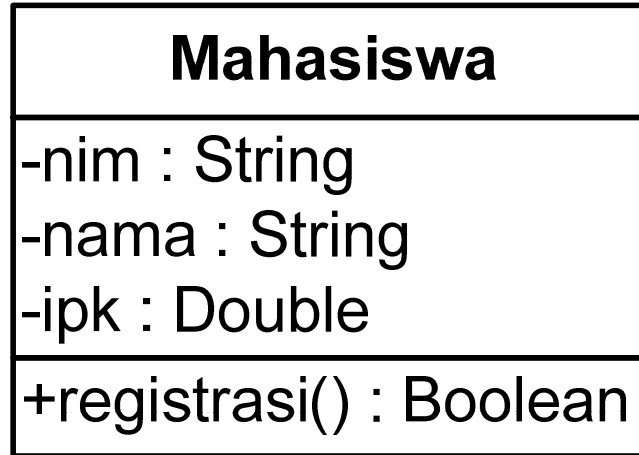
Construct	Description	Syntax
realization	a relationship between a specification and its implementation.	

Class Diagrams

- Notasi class digambarkan dengan kotak seperti contoh berikut:



Class Diagrams



- Modifier akses dilambangkan dengan 3 macam notasi:
 - (+) → public
 - (-) → private
 - (#) → protected

Class Diagrams

- Notasi dari atribut
 - visibility name: type multiplicity = default {property-string}
- Contoh
 - - name: String [1] = "Untitled" {readOnly}
 - + berarti public, - berarti private, # berarti protected
 - “Untitled” adalah nilai yang diberikan secara default jika tidak ditentukan saat objek dibuat
 - {readOnly} adalah properti tambahan dari atribut, dimana disini berarti tidak bisa dimodifikasi

Class Diagrams

- Notasi dari operations
 - visibility name (parameter-list) : return-type {property-string}
- Parameter pada parameter-list dinotasikan seperti pada atribut
 - direction name: type = default value
 - direction bisa berupa: in, out, atau inout
- Contoh
 - + balanceOn (in date: Date) : Money
- Bagaimana dengan constructor?
 - Sama dengan methods
 - visibility name_constructor([parameter-list])

Atribut dan Method

Shuttle
weight: Integer age: Integer status: enum = on-ground

Mission
start: Date end: Date cost: Dollars

Shuttle
start_engines() stop_engines() fuel_level(): integer launch(t:time)

Class Diagrams

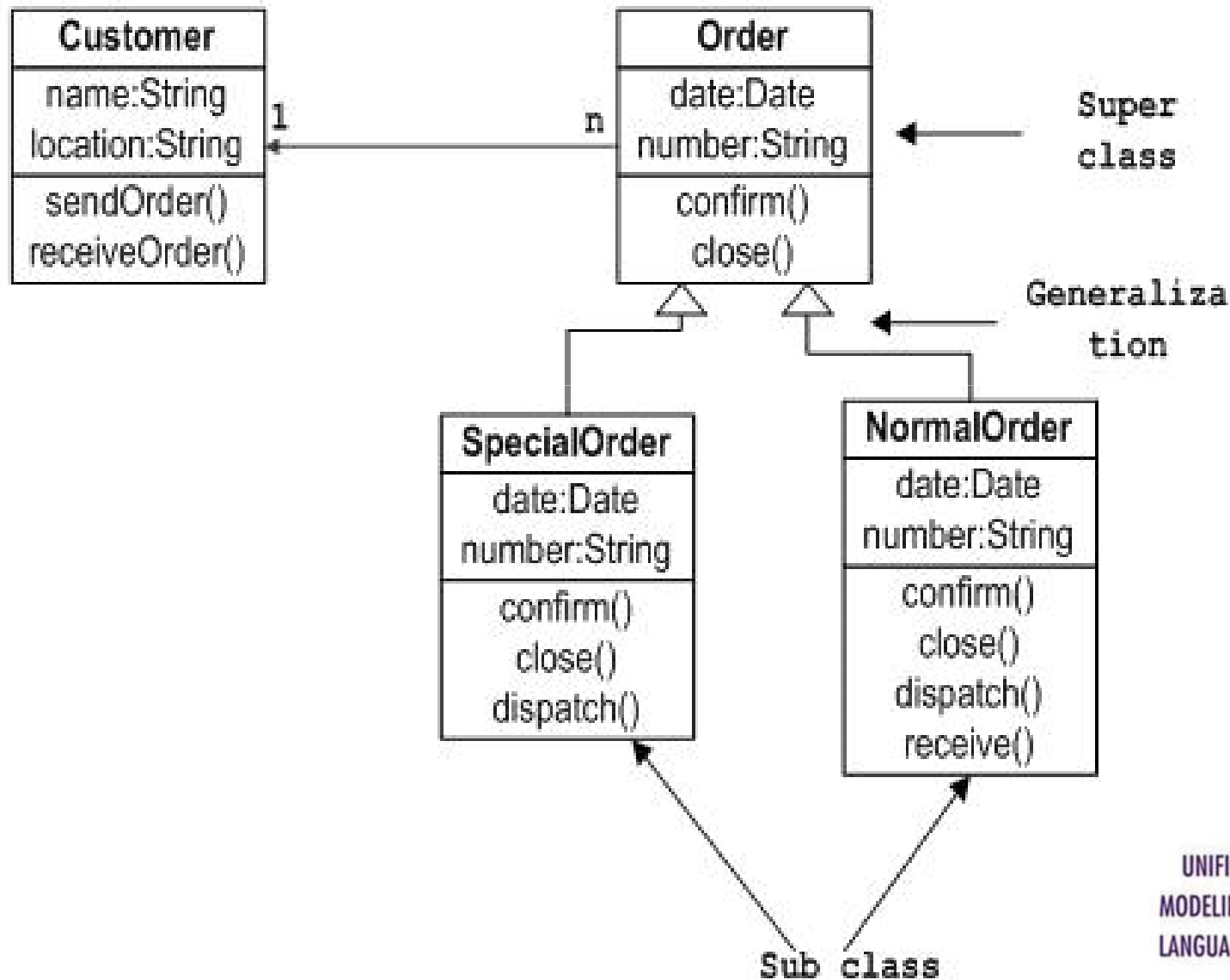
- Contoh: Buatlah class diagram dari program java di bawah ini:

```
class Anjing {  
    public Anjing() {  
        System.out.println("Anjing tercipta");  
    }  
    public String bersuara() {  
        System.out.println("Guk guk guk");  
    }  
    public void makan(String makanan) {  
        System.out.println("Sedang makan "+makanan);  
    }  
}
```

Class Diagrams

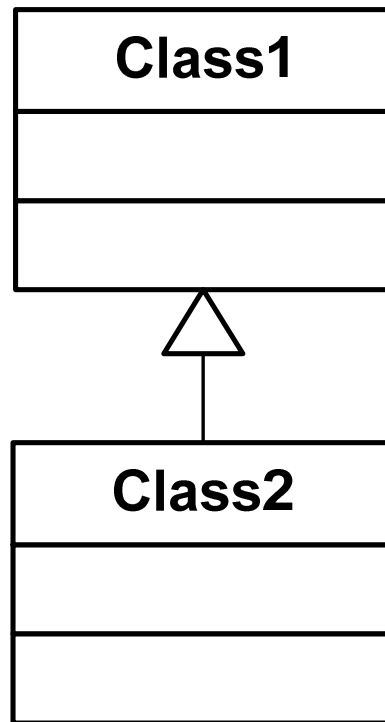
- Bagaimana kalau class tersebut merupakan abstract class?
- Bagaimana dengan abstract methods?
- Bagaimana kalau ada beberapa kelas dimasukkan dalam satu package?
- Bagaimana dengan interface?

Class Diagrams Example

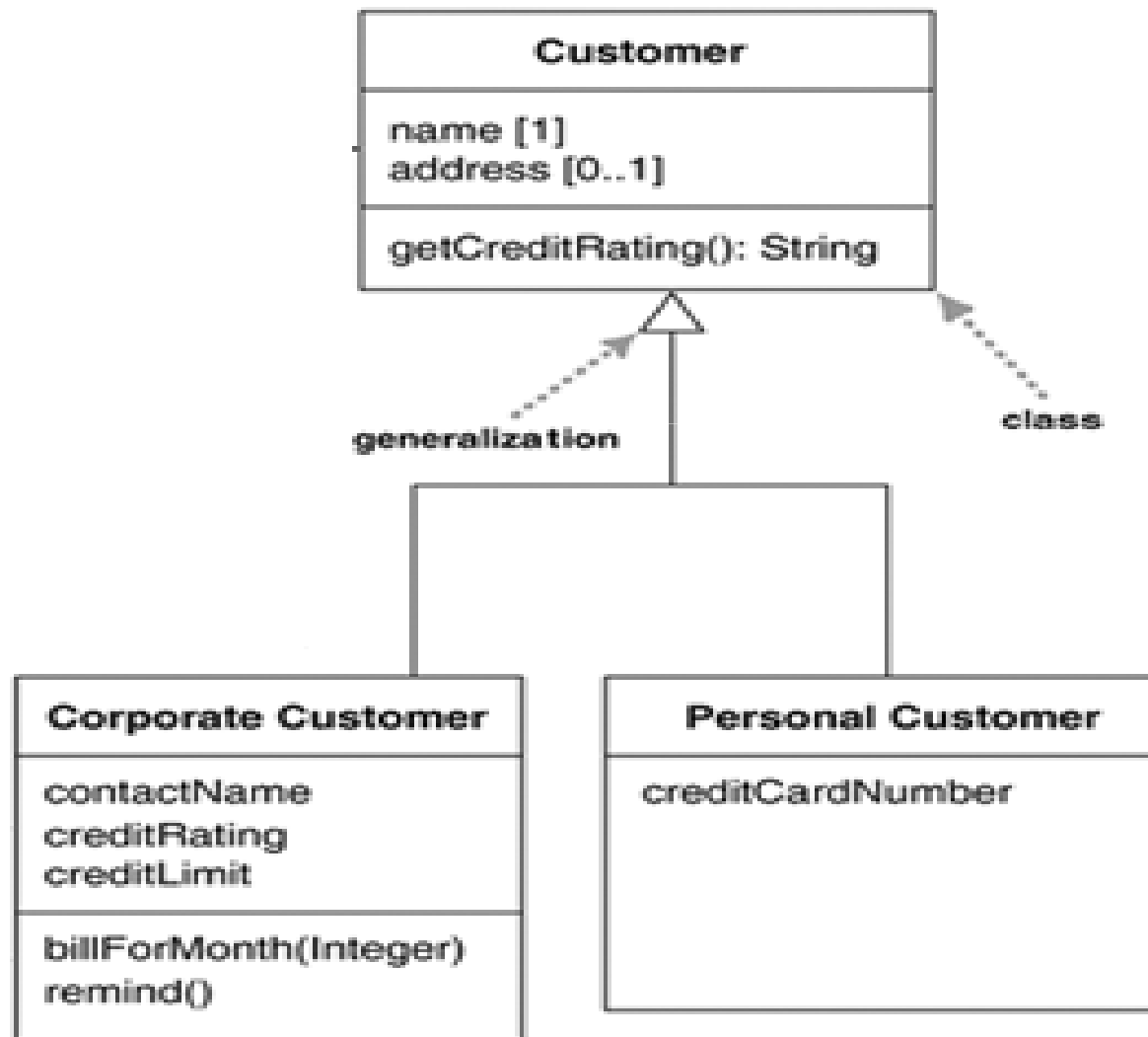


Class Diagrams

- Bagaimana kalau ada hubungan antar class seperti inheritance, aggregation, composition, dsb.?
- Inheritance:

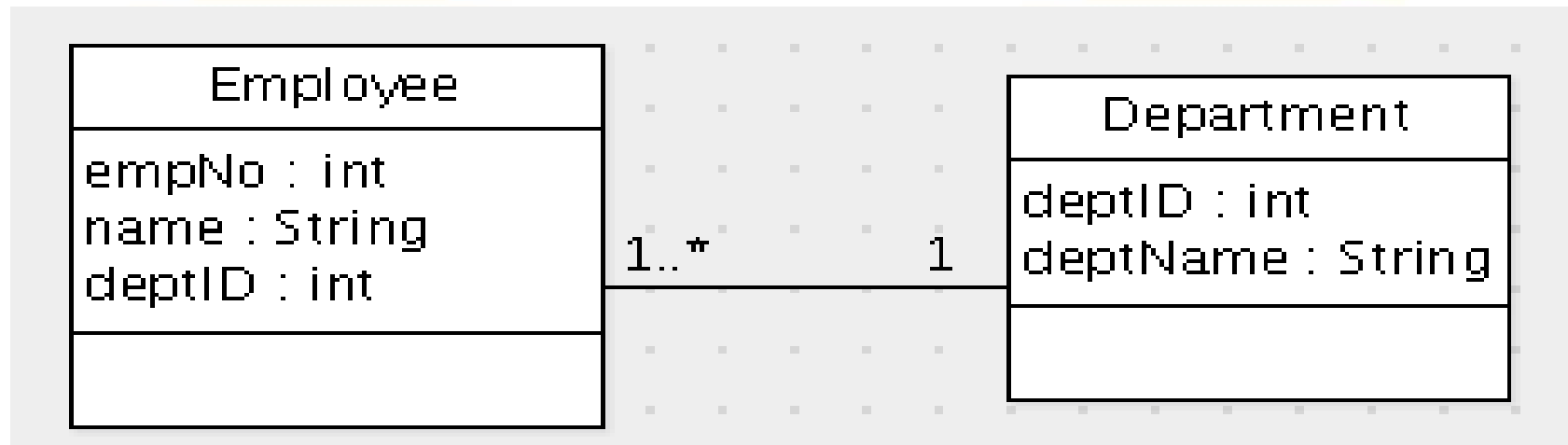


Generalization



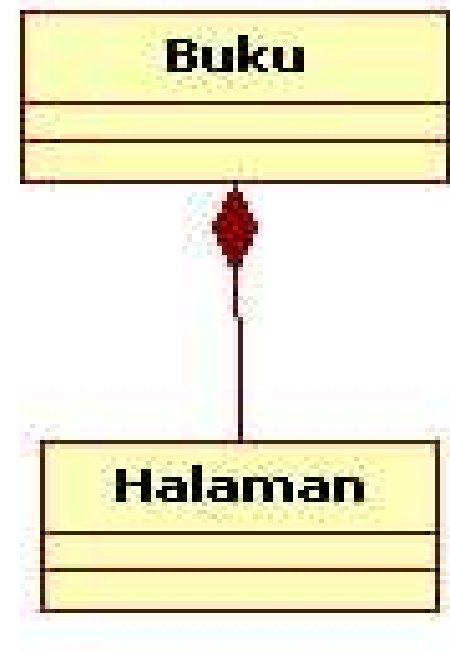
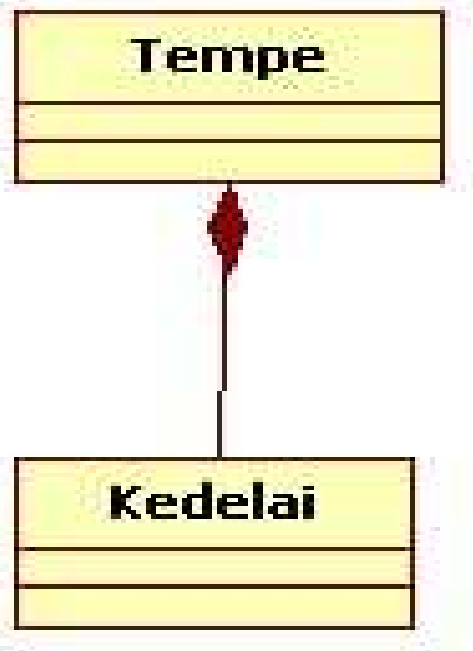
Class Diagrams

- Association:



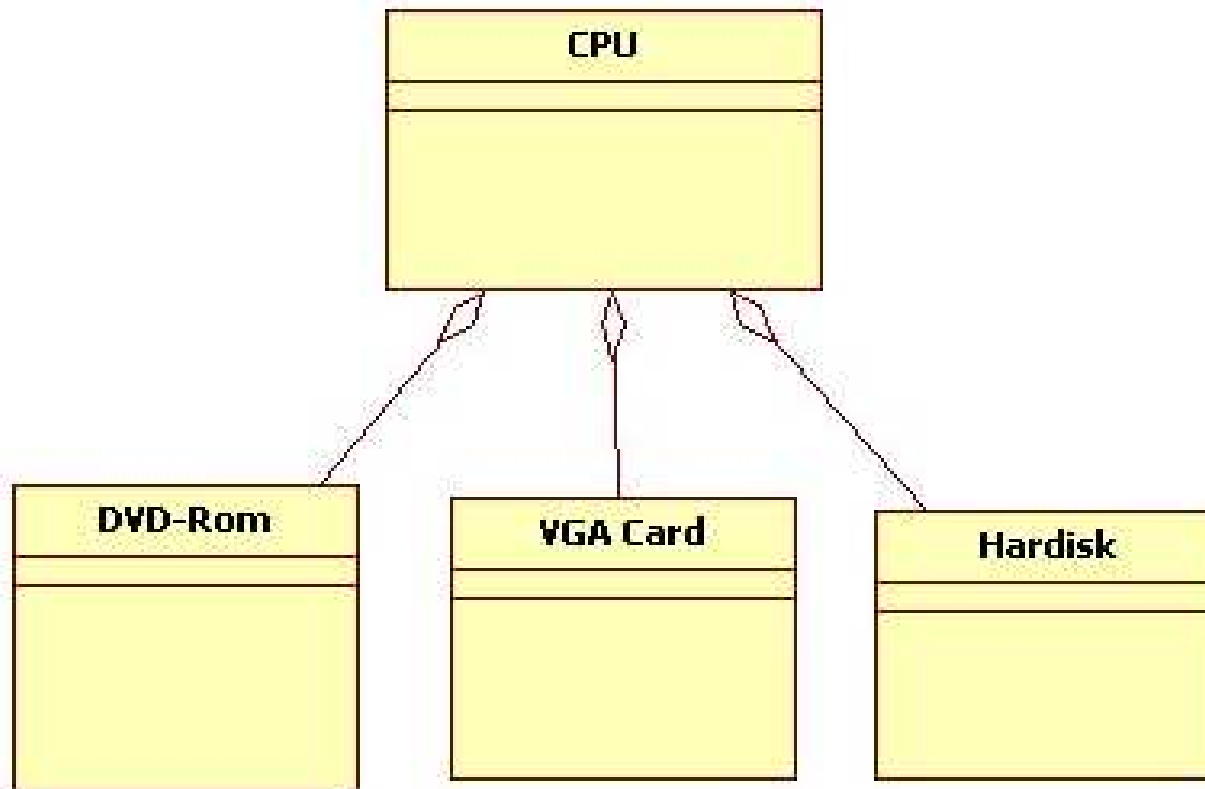
Class Diagrams

- Composition:



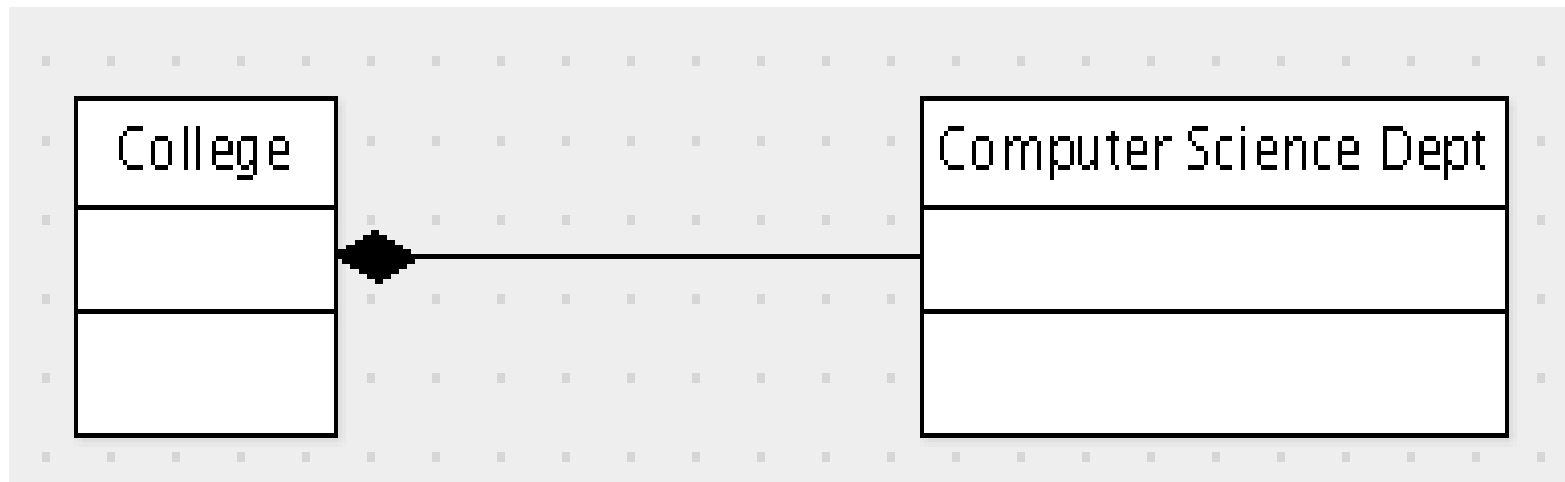
Class Diagrams

- Whole-part (aggregation):



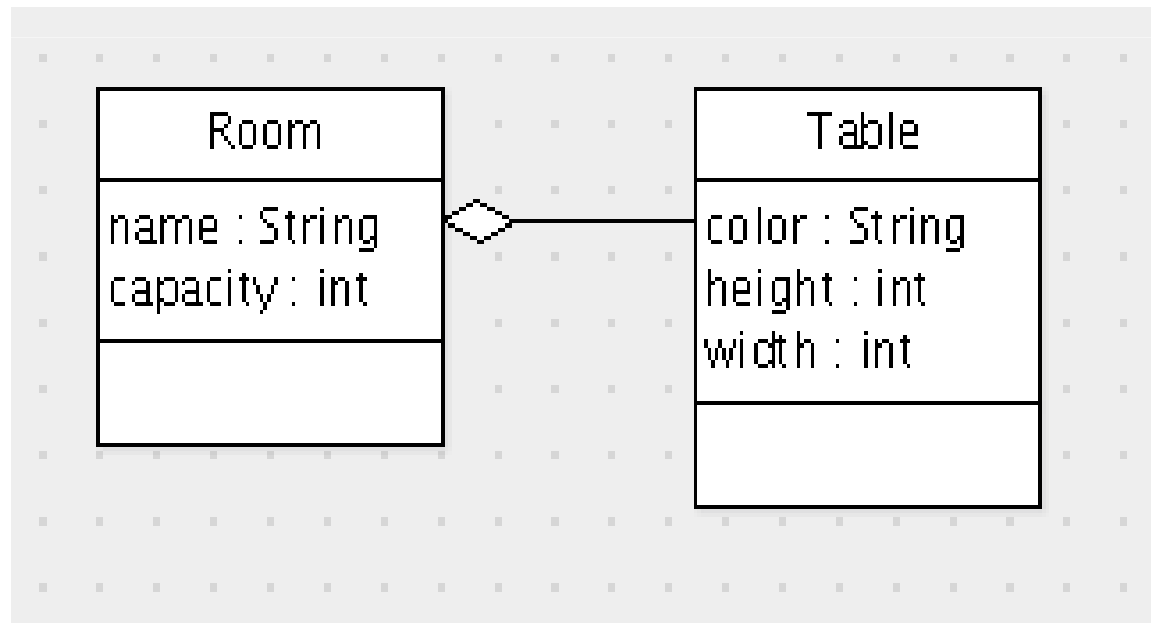
Composition

- 'has a' or 'contains a' relationship (whole-part)
 - Kampus memiliki fakultas CS atau kampus terdiri dari fakultas CS (salah satunya)
 - Jika tidak ada fakultas, tidak mungkin ada kampus

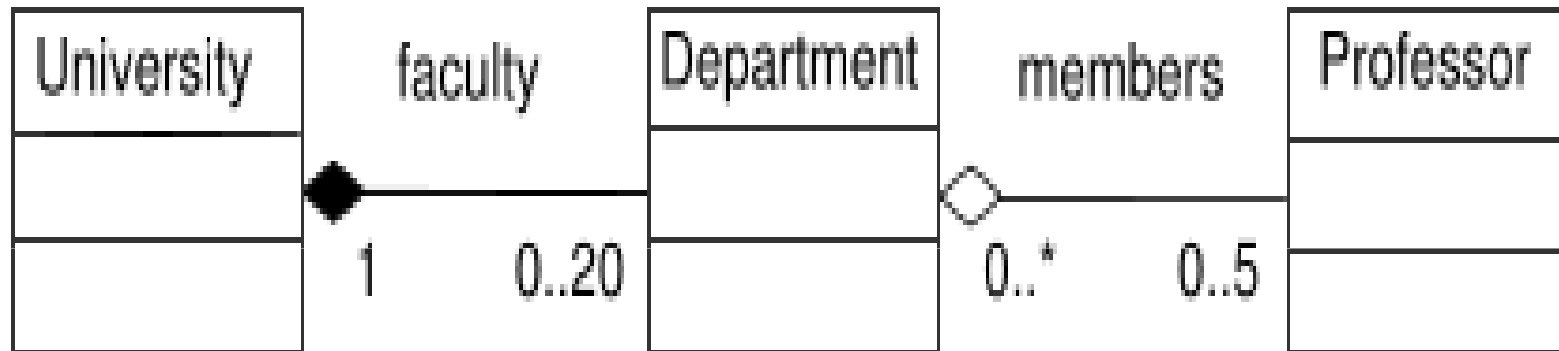


Aggregation

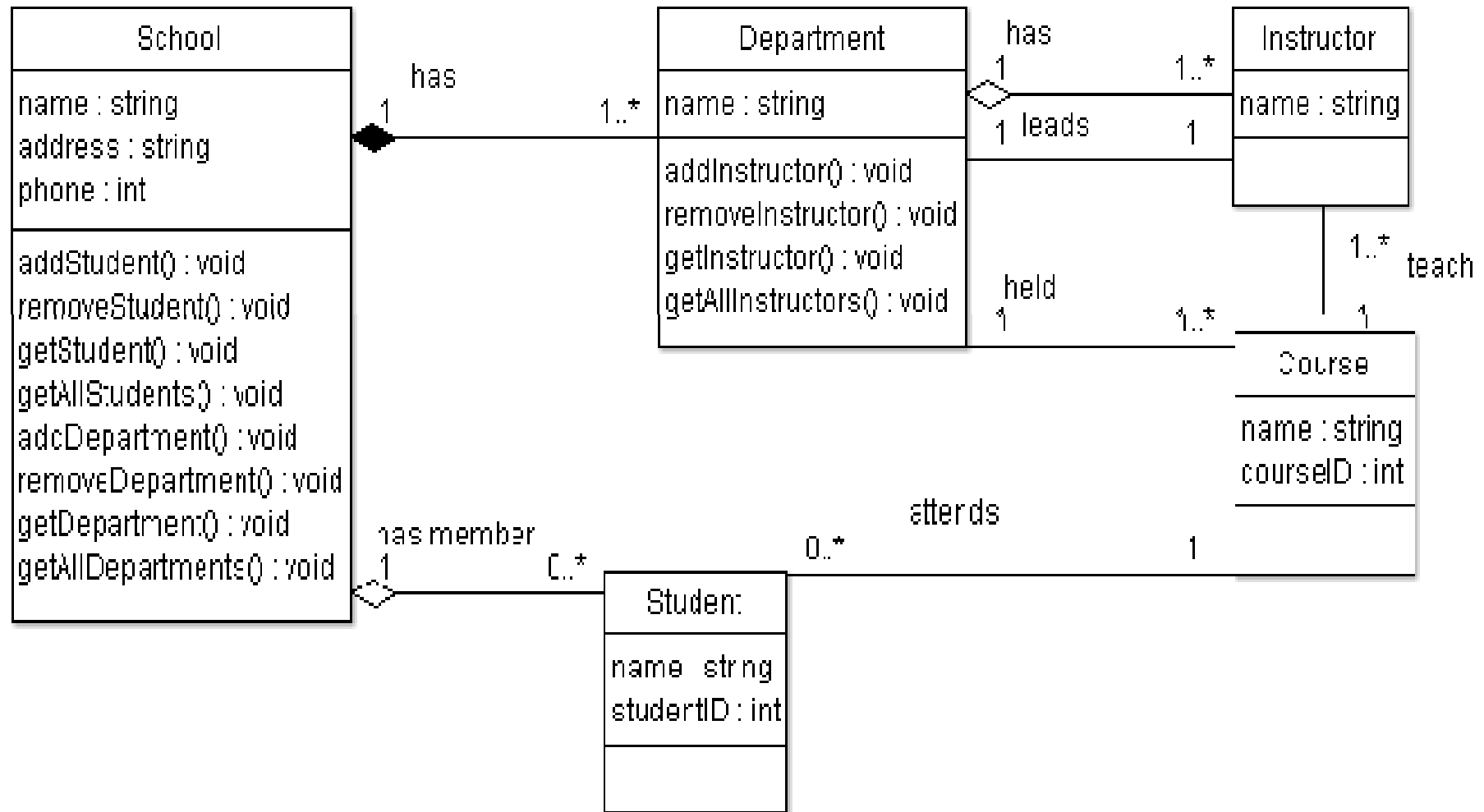
- Sebuah ruangan memiliki meja dan kursi
- Tanpa kehadiran ruang, meja dan kursi bisa tetap ada

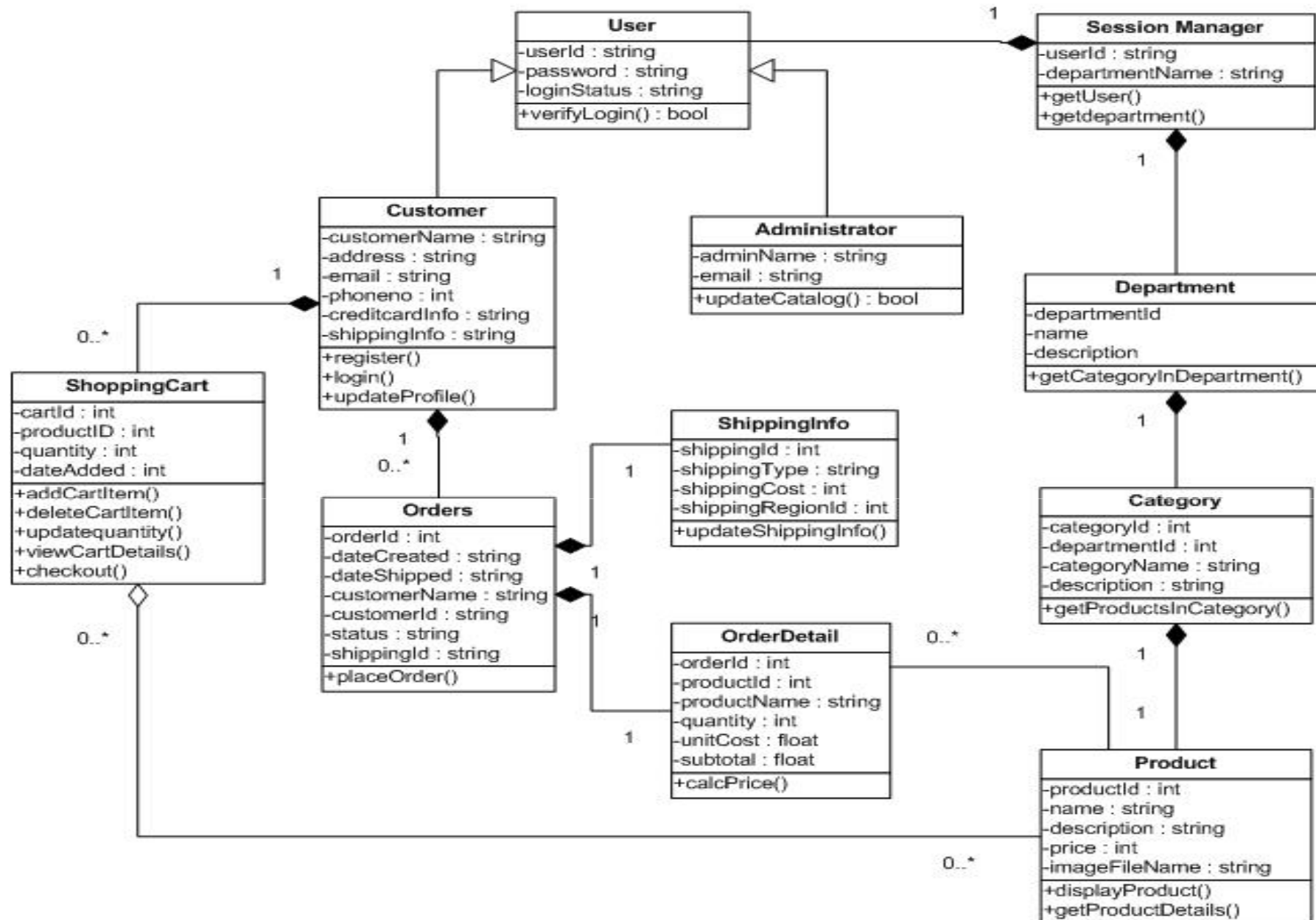


Class diagram Example



Class diagram Example





NEXT

- Exception Handling in Java